Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (previously presented): A computer implemented method for rearranging a computer program comprising:

organizing the computer program logically into a plurality of blocks;

constructing a dependency graph based on the organization of the plurality of blocks in the computer program;

determining a critical section included in the dependency graph;

detecting a portion of the plurality of blocks in the computer program that could be executed outside of the critical section;

inserting a plurality of dependency relationships based on the dependency graph between the plurality of blocks to cause execution of the detected portion of the plurality of blocks in the computer program outside of the critical section; and

rearranging the detected portion of the plurality of blocks to outside the critical section that were inside the critical section based on the inserted plurality of dependency relationships.

Claim 2 (previously presented): The method of claim 1, wherein the plurality of blocks includes computer program instructions.

Claim 3 (previously presented): The method of claim 1 further comprising organizing the plurality of blocks in the computer program based on a node and a super block, wherein the node includes a plurality of blocks and the super block includes a plurality of nodes.

Claim 4 (previously presented): The method of claim 1, wherein the critical section included in the dependency graph accesses shared resources.

Claim 5 (cancelled)

Claim 6 (previously presented): The method of claim 1 further comprising adding a termination point to the critical section if a portion of the critical section is outside of the dependency graph.

Claim 7 (previously presented): The method of claim 1 further comprising inserting an additional dependency relationship based on a direct dependency, an indirect dependency, or a shortest life-time dependency.

Claim 8 (previously presented): The method of claim 1 further comprising scheduling to execute the plurality of blocks in the computer program based on the dependency graph, after rearranging the detected portion of the plurality of blocks to outside the critical section.

Claim 9 (previously presented): A computer implemented system for rearranging a computer program comprising:

a computer program organizer, to organize the computer program logically into a plurality of blocks;

a dependency graph construction module, to construct a dependency graph based on the plurality of blocks of the computer program;

a critical section determination module to determine a critical section included in the dependency graph;

a detection module to detect a portion of the computer program recognized outside of the critical section that could be executed by the processor; and

a dependency relationships inserter, to insert a dependency relationship is between the plurality of blocks to cause execution of the detected portion of the computer program outside of a critical section.

Claim 10 (cancelled)

Claim 11 (previously presented): The system of claim 9, wherein the critical section included in the dependency graph accesses shared resources.

Claim 12 (previously presented): The system of claim 11, wherein the dependency relationships inserter inserts a termination point to the critical section blocks if a portion of the critical section is outside of the dependency graph.

Claim 13 (previously presented): A system for processing a plurality of network packets comprising:

a network processor;

a network interface to control the transmission between the network processor and a network;

a shared resource accessible to the plurality of network packets;

a network processor program to process the plurality of network packets;

a dependency graph constructor to construct a dependency graph based on the network processor program; and

a dependency relationship inserter to optimize the network processor program by inserting a plurality of dependency relationships based on the dependency graphs to rearrange the order in which the network processor program is executed.

Claim 14 (previously presented): The system in claim 13, wherein the dependency graph constructor further determines a critical section of the plurality of network packets and includes the critical section in the dependency graph.

Claim 15 (currently amended) The system in claim 13, wherein the dependency relationship inserter module inserts additional dependency relationships based on ~~a direct dependency,~~ an indirect dependency~~, or a shortest life-time dependency~~.

Claim 16 (previously presented): A non-transitory computer-readable storage medium that provides instructions that, when executed by a processor, causes the processor to:

organize a computer program logically into a plurality of blocks;

construct a dependency graph based on the organization of the plurality of blocks in the computer program;

determine a critical section associated with the dependency graph;

detect a portion of the plurality of blocks in the computer program that could be executed outside of the critical section;

insert a plurality of dependency relationships between the plurality of blocks to cause execution of the detected portion of the plurality of blocks in the computer program outside of the critical section; and

rearrange the detected portion of the plurality of blocks to outside the critical section that were inside the critical section based on the inserted plurality of dependency relationships.

Claim 17 (previously presented): The non-transitory computer-readable storage medium of claim 16, wherein the critical section included in the dependency graph accesses shared resources.

Claim 18 (previously presented): The non-transitory computer-readable storage medium of claim 16 further comprising instructions that insert a termination point to the critical section if a portion of the critical section is outside of the computer program.

Claim 19 (currently amended): The machine readable medium of method 16 further comprises inserting dependency relationships based on ~~a direct dependency, an indirect dependency, or~~ a shortest life-time dependency.

Claim 20 (new): The method of claim 1 comprising:

assessing whether a portion of the critical section is outside of the dependency graph; and

adding a termination point to the critical section and inside the dependency graph when the portion of the critical section is outside of the dependency graph.

Claim 21 (new): The method of claim 6 comprising inserting a first dependency relationship into and based on the dependency graph, between the plurality of blocks and the added termination point, to form a reconstructed dependency graph to cause execution of the added termination point prior to all other nodes in the reconstructed dependency graph.

Claim 22 (new): The method of claim 6 comprising inserting a first dependency relationship into and based on the dependency graph, between the plurality of blocks and the added termination point, to form a reconstructed dependency graph to cause execution of the added termination point subsequent to all other nodes in the reconstructed dependency graph.